About functions where function input describes inner working of the function

Rade Vuckovac rade.vuckovac@gmail.com

July 28, 2010

Abstract

This paper argues an existence of a class of functions where function own input makes function description. That fact have impact to the wide spectrum of phenomena such as negative findings of Random Oracle Model in cryptography, complexity in some rules of cellular automata (Wolfram rule 30) and determinism in the true randomness to name just a few.

1 Introduction

This paper is a continuation of research done on MAG type algorithms [1]. Although MAG algorithms are based on the use of the McCabe conditional complexity, the underlying complexity mechanism was never explored further. In this attempt we try to merge algorithms and composite functions, and also we point out the missing ingredient in defining a function's description.

2 The Mechanism behind complexity in systems with high conditional complexity

2.1 Conditional complexity

The McCabe [2] work on software testing is very relevant here, because it relates graph theory and algorithm input / output. The essence of this work is to treat an algorithm branching as a tree structure introducing the phenomena of conditional complexity. The conditional complexity is a metric or a number of independent paths through some software module. For example, if a source code of an algorithm does not have if / else algorithm structure then there is only one path through the source code. If there is one if / else then there are two paths through source code, one path when the conditional argument is TRUE and the other when the conditional argument is FALSE. See the figure 1.
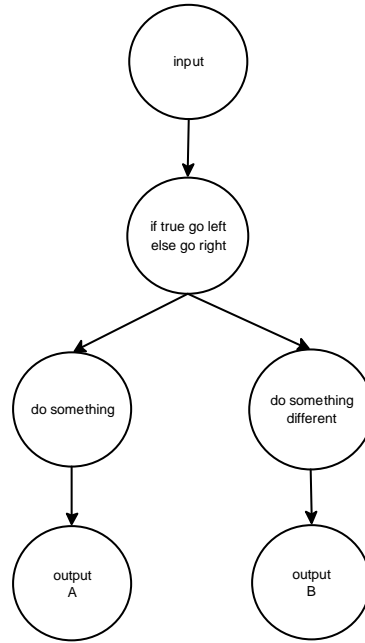
Figure 1:

Consequently, the if / else statement will double the number of possible paths through the source code, making a software module more complex. Therefore there is potential for exponentially more independent output cases for testing.
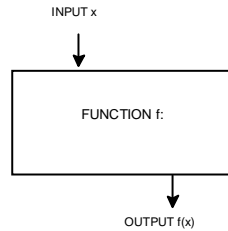
It is an interesting observation of McCabe that small programs (in lines of code terms) usually make complex ones because of the inclusion of several or more if/else programming compositions.

2.2 An algorithm as a function

The whole idea of conditional complexity can be taken a step further. Usually a function is given by a formula or a plot, or it is computed by an algorithm. The figure 2 from Wikipedia gives a general idea of a function. The next logical step is to treat an algorithm as a function, not just as a tool to compute a function. Certainly an algorithm can be a function in own right. For example, let us consider the algorithm from figure 3. Like a function, here we have an input and an output. The "Black box" of the function in principle can be anything; in this case it is an algorithm (the same idea as figure 2).

2.3 3n+1 problem (the Collatz conjecture)[3]

> ... The Collatz conjecture is an unsolved conjecture in mathematics. It is named after Lothar Collatz, who first proposed it in 1937.

2

Figure 2:

INPUT x

FUNCTION f:

OUTPUT f(x)

...A function f takes an input, x, and returns an output f(x).
One metaphor describes the function as a "machine" or "black box" that converts the input into the output...
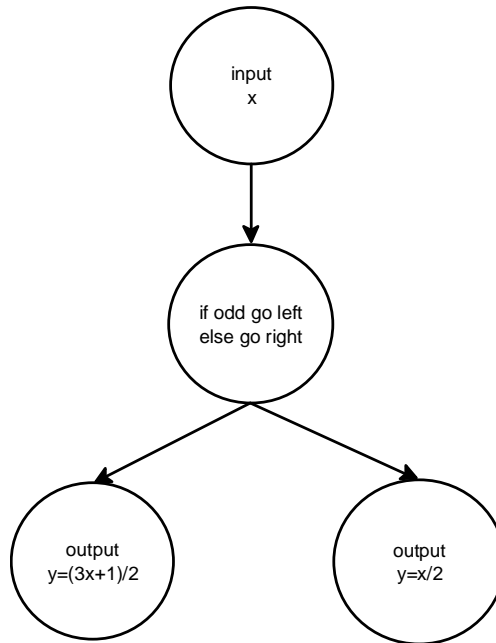Wikipedia

Figure 3:

input
x

if odd go left
else go right

output
y=(3x+1)/2

output
y=x/2

The conjecture is also known as the 3n + 1 conjecture, as the Ulam conjecture (after Stanislaw Ulam), or as the Syracuse problem; the sequence of numbers involved is referred to as the hailstone sequence or hailstone numbers, or as wondrous numbers per Gödel, Escher, Bach. We take any number n (element of N+). If n is even, we halve it (n/2), else we do "triple plus one" and get 3n+1. The conjecture is that for all numbers this process converges to 1. Hence it has been called "Half Or Triple Plus One", sometimes called HOTPO. Paul Erdős said about the Collatz conjecture: "Mathematics is not yet ready for such problems." He offered $500 for its solution. (Lagarias 1985) ...

Experimental evidence The conjecture has been checked by computer for all starting values up to 10 × 2^58 ≈ 2.88×10^18. While impressive, such computer evidence should be interpreted cautiously.
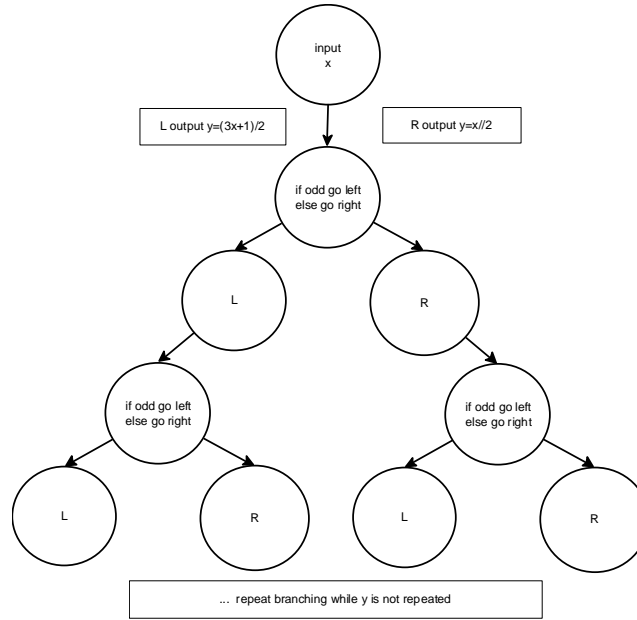
Figure 4:

More than one important conjecture has been found false, but only
with very large counterexamples. (See for example the Pólya conjec-
ture, the Mertens conjecture and the Skewes' number.) ... (Wikipedia)

Interestingly enough our figure 3 with little massaging is exactly the 3n+1 prob-
lem. (see figure 4). It should be noted that this algorithm does not stop nec-
essarily on y = 1. It will stop when the y value is repeated or in other case
it will branch forever (which is also interesting prospect). ***Here we can say
that there is the set of numbers of all checked numbers so far (up to
10×2^{58}) that obeys the Collatz conjecture and we will call that set S***.
Also we observe that all members of that set are mapped to 1 by the algorithm
/ function as shown in figure 4. Now our figure 4 almost resembles the diagram
of a composite function as shown in figure 5.

***The fundamental difference between figure 4 and 5 is that a composite
function defines strictly one path only. That is a contrast to our
figure 4 which leaves the composition of a particular path open.*** In both
cases, the processes are deterministic. But in the case of figure 4 the algorithm
/ function is not fully described. According to the conditional complexity the
input may went through any possible path towards the output. And if we
enumerate left branching as f and right branching as g, for first two levels of
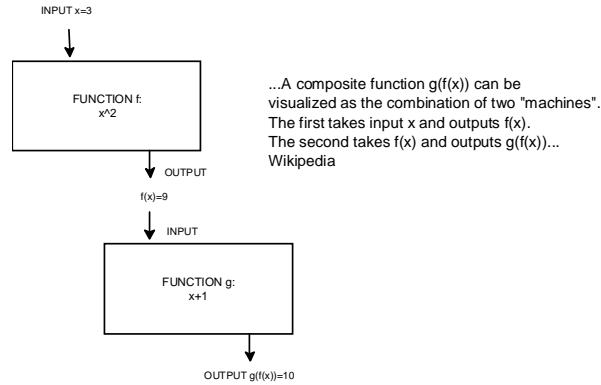branching in figure 4 we have 4 combinations of f and g:

4

INPUT x=3

FUNCTION f:
x^2

...A composite function g(f(x)) can be
visualized as the combination of two "machines".
The first takes input x and outputs f(x).
The second takes f(x) and outputs g(f(x))...
Wikipedia

OUTPUT

f(x)=9

INPUT

FUNCTION g:
x+1

OUTPUT g(f(x))=10

Figure 5:

combinations f,g
$f \circ f$
$f \circ g$
$g \circ g$
$g \circ f$

We also know that order of execution is important because if we start with function g instead of f (figure 5) we will have different output (meaning different mapping). That means we have 4 distinct mapping for first two levels of branching (figure 4) and depending on input value one of 4 will be chosen for "input to output" transformation. It can be noted that number of unique arrangements of f's and g's will exponentially rise with every consequent branching. The empirical evidence of Collatz conjecture confirms the uniqueness of every mapping because for every known input the outcome will converge to 1.

To fully describe the algorithm / function from figure 4, i.e. to enable determination of the argument though calculation, every conditional decision must be recorded: for instance every left turn with 'L' and every right turn with 'R'. The encoding of the description of the particular mapping thus will be a string of L's and R's. That string can serve as a full description of particular composite function for particular input (to map amount and order of f's and g's. For two levels of branching from figure 4 the four possible strings would be:

combinations f,g	makes
$f \circ f$	LL
$f \circ g$	LR
$g \circ g$	RR
$g \circ f$	RL

The same description (L's R's string) can be used to recompose the inverse steps through the algorithm (stepping from y to x).

Now we come to the important question of a function's description. Algorithmic information theory, and particularly the Kolmogorov – Chaitin complexity
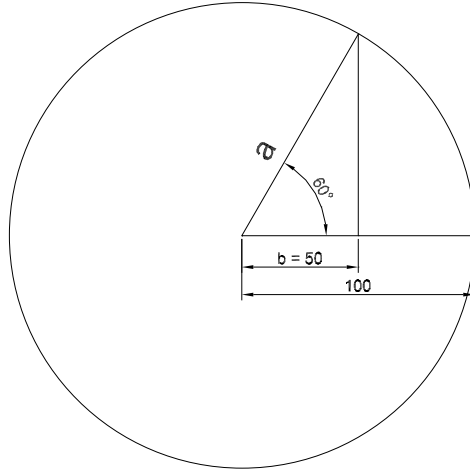
5

Figure 6:

paradigm[4] will use the algorithm from figure 4 as a measurement of complexity and therefore decide that the complexity of mapping is relatively low, because whole algorithm can be coded in tens lines of code. On the other hand there is considerable difficulty ("Mathematics is not yet ready for such problems." Paul Erdős) to find any formal way to predict the function output of figure 4.

To illustrate a "prediction quality" of classical functions figure 6 can be used. We can see that b is dependent on angle, so we can say with certainty that if 'a' is rotated in clock wise direction (making angle less than 60 degrees but still more than 0 degrees) then 'b' will lay somewhere between 50 and 100. In contrast, for the case of the figure 4 we have 10×2^{58} inputs converging to 1, but there is no way excluding intuition (in other words there is no formal way) of predicting the output of non-tested input. That simply means that if we really want to fully describe the 3n+1 function we have to include the description of every input path transformation (L's R's strings).

That also means, for some cases the Kolmogorov – Chaitin complexity is not entirely accurate because the algorithm description does not mean full description of a phenomena as in figure 6 but may be more as in figure 4.

From a skeptical point of view it can always be argued that set of recorded L's R's strings may be compressed in some way (perhaps finding some pattern) and still keep the relatively low complexity of 3n+1 function's description. If we assume that this is the case, then our L's R's descriptions for every input may be shorter than required binary representation for all inputs. That is unlikely, because L's R's descriptions entropy can not be smaller than Shannon's entropy.

Let say that b (in our case b = 58) is the number of bits required to describe every input of the set S and r is the number of bits required to describe all L's R's recorded strings for the set S. Because our L's R's string is in the same time the description of every path and consequently the encoding of every input, we can deduce that:

$b \leqq r$

That means that the complexity of the full function description (figure 4, set S) can not be reduced below the complexity of input (set S).

It should be noted that there were attempts to prove that the Collatz conjecture is unprovable (see [5]). While argumentation about unprovable concerns the set of all natural numbers, in this paper argumentation are within our set S (empirically checked numbers converging to 1).

2.4   Stephen Wolfram's rule 30

The rule 30 belongs to class 3 behavior cellular automata (CA). This class has a complex structure and can be regarded as a chaotic/random class.

Rule 30 [p869 [6]] can be expressed graphically as figure 7. The English formulation is [p27 [6]]: "First, look at each cell and its right-hand neighbor. If both of these were white on the previous step, then take the new colour of the cell to be whatever the previous colour of its left-hand neighbor was. Otherwise, take the new colour to be the opposite of that."

It is obvious that all complexity obtained by this algorithm is a consequence of the McCabe conditional complexity which is evident from the above rule (if something is TRUE do something, else do something different, and the output is next input for the repetition of if / else, very similar to the figure 4).

There is also an explanation for the same apparent randomness outcome (for rule 30) for a start with high entropy and for a start with low entropy (see figure 8[figure 8 [6] p281]).

The author rightly argues that initial conditions does not play a role in randomness development for rule 30, opposite to classical chaotic systems in which small changes in the initial conditions may strongly effect the outcome. However, the author does not explain cause of apparent randomness, but now we know that conditional complexity is indifferent to input entropy and depends solely on the amount of branching: after so many branching steps the second picture in figure 8 ought to look as the first one after a while.
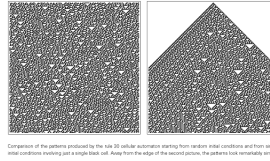
7

Figure 7:

Comparison of the patterns produced by the rule 30 cellular automaton starting from random initial conditions and from simple initial conditions involving just a single black cell. Away from the edge of the second picture, the patterns look remarkably similar

Figure 8:

2.5   Random Oracle Methodology

Our current understanding of randomness also may be revised. For example we have statements like "Anyone who considers arithmetical methods of random digits is, of course, in a state of sin."[7] . It will look benign if exchange 'arithmetical methods' with 'deterministic methods' because of our perception on randomness, but then above statement is not correct.

Loosely, the formal notation of the above statement is developed through Random Oracle Methodology(ROM)[8] and the methodology's negative results claim that the concrete hash function cannot be substituted for the random oracle. A problem with this result is that in practice we have hash functions which are secure for apparently no obvious reason. Some advancement in that problem can be made in redefining what entitlements have the term "fully described functions", because it appears that the term "fully described function" in ROM goes along lines from figure 6. Consequently that assumption is false because we have the case from figure 4.

On the other hand ROM results show that if complexities of a function's description and complexities of inputs are the same then the function is indistinguishable from random oracle,, although that scenario is dismissed on practicality grounds. Again, the algorithm from figure 4 clearly satisfies requirement of description and input complexity being the same. Also calculating an instance of the algorithm from figure 4 is more than practical. Following ROM argumentation we have a deterministic process with output which can not be distinguished from true randomness.

There may be some pointers for a design of hash functions and ciphers. The complexity in cryptography is mainly acquired by focusing on complexity of

8

individual functions and functions predefined compositions (in principle same concept as figure 5). The alternative may be to make functions (f's and g's) simple but to compose them in dynamical and unpredictable way (applying high conditional complexity figure 4) therefore achieving requirement for the same complexity of the function inputs and the full function description

3   Conclusion

The following is concluded:

1. An algorithm can definitely be considered as a form of a composite function.

2. Giving the description of an algorithm does not mean giving a full description of the composite function, but does mean that the algorithm is deterministic.

3. There are some cases where the full function's description complexity and the input complexity coincide while the underlying algorithm is fairly simple and easy to execute.

From an algorithm perspective, there is a formulation from Bohm and Jacopini's work [9]which demonstrates that all programs could be written in terms of only three control structures: (a) The sequence structure, (b) The selection structure and (c) The repetition structure. It can be argued that formalism may not be relevant when '(b) the selection structure' is used in an algorithm causing conditional complexity. In other words we can have deterministic processes which may not be possible to formally distinguish from the true randomness.

References

[1] http://www.ecrypt.eu.org/stream/mag.html

[2] http://www.literateprogramming.com/mccabe.pdf

[3] http://en.wikipedia.org/wiki/Collatz_conjecture

[4] http://en.wikipedia.org/wiki/Kolmogorov_complexity

[5] Craig Alan Feinstein: arXiv:math/0312309v16 [math.GM]

[6] http://www.wolframscience.com/thebook.html

[7] John von Neumann (quoted in Knuth, Vol.2)

[8] http://eprint.iacr.org/1998/011.pdf

[9] http://en.wikipedia.org/wiki/Structured_program_theorem